**Laboratorio #4**

*Version #2*

Ejercicio #1

*Programa adjunto: e1.c*

*Ejecutando el programa:*

A screen shot of a computer program

Description automatically generated

Ambos procesos se encuentran en el estado en el que están "intentando adquirir" recursos, lo que sugiere que ambos están bloqueados esperando que el otro libere el recurso que necesitan. En este caso, el proceso 1 ha adquirido el recurso 1 y está esperando el recurso 2, que está bloqueado por el proceso 2, y viceversa. Por lo tanto, ambos procesos están en un estado de deadlock, lo que significa que ninguno de ellos puede continuar y el programa queda atrapado en un bucle infinito de espera.

*Describe cómo funciona un algoritmo de detección de deadlock y cómo se relaciona con la  
concurrencia en sistemas operativos.*

* Los algoritmos de detección de deadlocks son mecanismos utilizados por los sistemas operativos, que examinan el estado actual del sistema, específicamente los recursos asignados y las solicitudes pendientes que hay, y con esta información determinan si existe un ciclo en el grafo de asignación de recursos. Cuando se encuentra un ciclo, se determina que hay un deadlock.
* Esto se relaciona con la concurrencia en los sistemas operativos ya que es la capacidad de ejecutar múltiples procesos simultáneamente, sin caer en problemas como los deadlocks, que son una preocupación imporante en los sistemas concurrentes ya que estos pueden ocurrir cuando varios procesos están compitiendo por el mismo recurso, y al darse, se termina en una pérdida de productividad o incluso la muerte del sistema completo, por el bloqueo.

*¿Qué estrategias podrían implementarse para prevenir deadlocks en sistemas concurrentes más  
complejos que el ejemplo proporcionado?*

* Asignación ordenada de los recursos, con un orden fijo que debe ser seguido por los procesos al solicitar sus recursos
* Asignación anticipada, donde los procesos solicitan todos los recursos que van a necesitar antes de su ejecución
* Liberación de recursos, los procesos tienen que liberar los recursos que ya no necesitan antes de pedir nuevos
* Exclusión mutua, garantizar que solo un proceso tenga acceso a un recurso en un momento específico

*Explica cómo se podría modificar el código para introducir una situación de interbloqueo más sutil  
que no sea tan evidente como la inversión de recursos.*

* Aumentando la cantidad de procesos y de recursos, y que en vez de ser una inversión, sea un proceso utilizando alguna función de randomización, lo que dificultaría la detección del deadlock. Por ejemplo si creamos patrones de solicitud y de liberación basados en una función random, se ejecutarían durante un mayor tiempo hasta que coincida el random con una inversión como la del ejemplo.

*¿Qué métodos de detección y resolución de deadlocks conoces y cómo se aplican en sistemas  
operativos modernos?*

* Problema de readers and writers:
  + Detección:
    - Los sistemas operativos modernos pueden utilizar algoritmos de detección de deadlock para identificar si existe un deadlock en el sistema causado por el problema de lectores y escritores. Esto implica monitorear el estado actual del sistema en términos de los recursos compartidos y las solicitudes de acceso, y detectar si hay un ciclo de bloqueo en el grafo de asignación de recursos.
  + Resolución:
    - Establecer una política de priorización de escritores sobre lectores. Esto significa que si un escritor está esperando para acceder al recurso compartido, ningún lector nuevo debe ser permitido para acceder hasta que el escritor haya completado su operación. Esto evita que los lectores bloqueen a los escritores indefinidamente, lo que podría llevar a un deadlock.
  + Aplicación en OS modernos:
    - Se puede abordar mediante políticas de control de acceso a recursos compartidos. Por ejemplo, una política que prioriza a los escritores sobre los lectores para evitar que los escritores queden bloqueados indefinidamente detrás de lectores. Esto puede lograrse dándole acceso exclusivo al recurso compartido a los escritores en lugar de permitir que los lectores lo compartan mientras un escritor está esperando.
* Problema del banquero:
  + Detección:
    - El sistema operativo mantiene información sobre los recursos disponibles y las solicitudes de recursos de cada proceso. Si una solicitud de recurso puede conducir a un estado inseguro (donde se produciría un deadlock), el sistema operativo puede rechazar la solicitud o suspenderla temporalmente para evitar el deadlock.
  + Resolución:
    - Una vez que se detecta un deadlock, el sistema operativo toma medidas para resolverlo. Esto podría ser la liberación de algunos recursos asignados a procesos involucrados en el deadlock o, en casos extremos, la terminación de uno o más procesos para romper el ciclo de interbloqueo y permitir que los demás procesos continúen su ejecución.
  + Aplicación en OS modernos:
    - En un sistema operativo moderno, el algoritmo del banquero puede implementarse dentro del planificador del sistema. Cuando un proceso solicita recursos, el planificador verifica si la asignación de recursos conduciría a un estado inseguro utilizando el algoritmo del banquero. Si la asignación es segura, se otorgan los recursos al proceso. Si no es segura, el proceso puede ser suspendido temporalmente hasta que los recursos estén disponibles o se liberen otros recursos para evitar el deadlock.